Capstone Project

Machine Learning Engineer Nanodegree

Anton Levashov 27/10/2017

# Introduction

## Project Overview

Cdiscount.com generated nearly 3 billion euros last year, making it France’s largest non-food e-commerce company. While the company already sells everything from TVs to trampolines, the list of products is still rapidly growing.

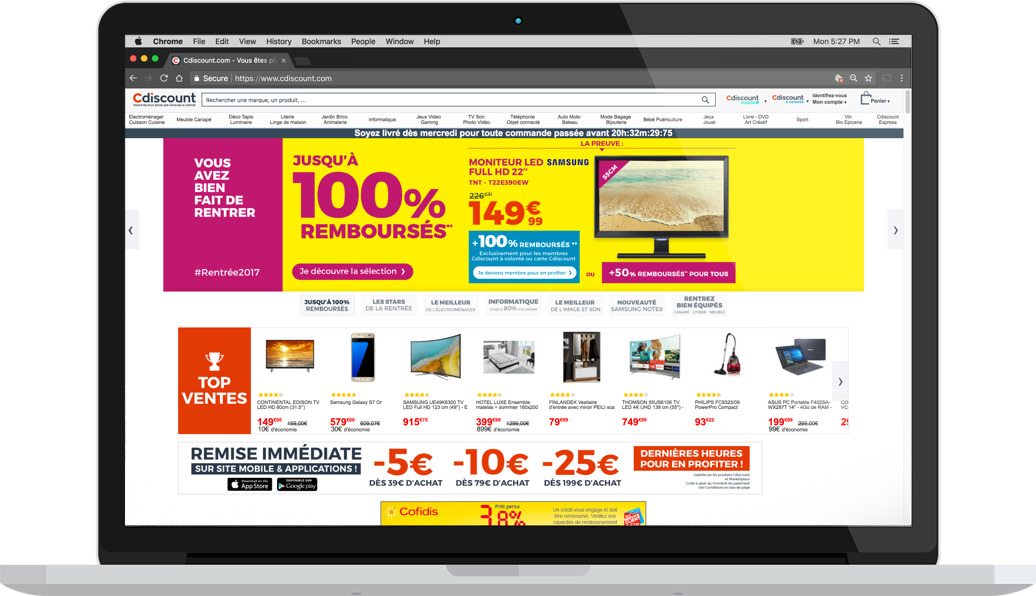


Figure 1 Cdiscount.com home page.

By the end of this year, Cdiscount.com will have over 30 million products up for sale. This is up from 10 million products only 2 years ago. Ensuring that so many products are well classified is a challenging task. Currently, Cdiscount.com applies machine learning algorithms to the text description of the products in order to automatically predict their category. As these methods now seem close to their maximum potential, Cdiscount.com believes that the next quantitative improvement will be driven by the application of data science techniques to images.

The purpose of this project [1] was to classify images in 5270 categories - predict the category of a product based on its image(s). The winning algorithm will help to automatically upload the new product in most suitable category just by looking at image of this product.

Ex: If the product is laptop battery the algorithm should put it the following category on the website: *INFORMATIQUE > CONNECTIQUE – ALIMENTATION > BATTERIE D'ALIMENTATION INFORMATIQUE*.

Cdiscount have a huge number of products on-line - roughly 18 million, this project will try to replace hand-labeling of the products by automatic labeling. The final step is to submit the winning solution to this competition and help Cdiscount to achieve the automatization of the labeling process.

## Problem Statement

The goal of this project was to find the best model that automatically classifies the products based on their images. As a quick tour of Cdiscount.com's website can confirm, one product can have one or several images. The data set Cdiscount.com is making available is unique and characterized by superlative numbers in several ways:

* Almost 9 million products: half of the current catalogue
* More than 15 million images at 180x180 resolution
* More than 5000 categories: yes this is quite an extreme multi-class classification!

In this project we used a certain type of Neural Networks - Convolutional Neural Networks (CNN) to classify the images in this project. CNNs are the best available technic for image classification/treatment in both performance and precision. Here is rough representation of CNN architecture:



Figure 1 Typical CNN architecture.

Training neural networks needs a large number of training examples; fortunately in this competition we had 7000000 images for training and 1700000 for the test sets! But we also used Data Augmentation technic to increase the size of our training/validation sets by taking each training image and creating multiple random transformations around the bounding box of the object using ImageDataGenerator from Keras. In this project we have use just some small transformation of original images (random crop and horizontal-flip).

But training this kind of networks can take a lot of hours and even days. [2]

To cut training time we have used Transfer Learning [3], it's an important concept for Deep Learning. In Transfer Learning we retrain the already trained models by giving the input as our own dataset. We can cut the size of data required for training and especially the training time. The training uses the weights from the already trained model and starts learning new weights in only the last fully connected layers.

## Metrics

The evaluation metric for this project was % of the categorization accuracy of the predictions of 5270 categories (the percentage of products the solution gets correct). Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

C:\Users\WMFP6357\Desktop\5.png

*Source* [*scikit-learn.org*](http://scikit-learn.org/stable/modules/model_evaluation.html)

We can observe the accuracy of other participant’s solutions on the public leader board in this competition, so it helps to understand if our solution performs better or worse compared to others. Currently the winning solution is 79.567% of accuracy.

# Data

## Data sets

This competition uses the real-world data sets, from current catalogue of products on Cdiscount. Files are in BSON format. BSON, short for Binary JSON, is a binary-encoded serialization of JSON-like documents, used with MongoDB.



Figure 2 A samples of images with respective category.

It's a challenging data set of over 15 million images of products representing 9 million products in over 5,000 categories. A product can have one or several images associated, each image has 180x180 resolutions in RGB.

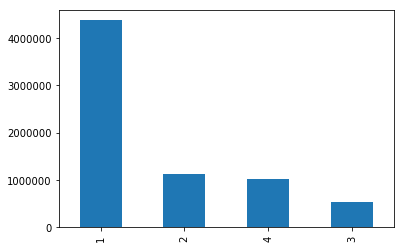


Figure 3 Distribution of number of images per product.

For every product \_id in the test set, we should predict the correct category\_id. Here is the [description of files](https://www.kaggle.com/c/cdiscount-image-classification-challenge/data) used in this project:

* train.bson - (Size: 58.2 GB) Contains a list of 7,069,896 dictionaries, one per product. Each dictionary contains a product id (key: \_id), the category id of the product (key: category\_id), and between 1-4 images, stored in a list (key: imgs). Each image list contains a single dictionary per image, which uses the format: {'picture': b'...binary string...'}. The binary string corresponds to a binary representation of the image in JPEG format. This kernel provides an example of how to process the data.
* test.bson - (Size: 14.5 GB) Contains a list of 1,768,182 products in the same format as train.bson, except there is no category\_id included. The objective of the competition is to predict the correct category\_id from the picture(s) of each product id (\_id). The category\_ids that are present in Private Test split are also all present in the Public Test split.
* category\_names.csv - Shows the hierarchy of product classification. Each category\_id has a corresponding level1, level2, and level3 name, in French. The category\_id corresponds to the category tree down to its lowest level. This hierarchical data may be useful, but it is not necessary for building models and making predictions. All the absolutely necessary information is found in train.bson.

## Data Exploration

The distribution of product group is highly skewed - there is a lot of products in one class and very few in another. Apparently the number of training examples (images) per class (category) varies a lot in the training set, from over 80000 in the classes with most training examples to just 12 in the class with the least.

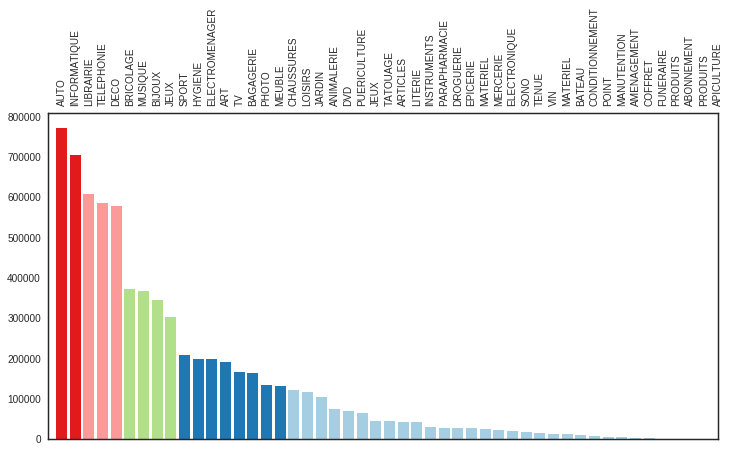


Figure 4 Number of occurrence for each product in categories.

So we can assume that the testing set has a different distribution and a class with just a few images in the training set can have thousands of images in the test set - imbalance problem is present.

Ex: the most frequent category found 79640 times is MUSIQUE and 31 less frequent categories are PUERICULTURE, APICULTURE, etc., are found only 12 times.

It looks like distribution of product category is highly skewed along product id. Most frequent product in all products is just 19th frequent among the first 100000 products.

# Solution Statement

## CCN Architecture

In this project we have used Convolutional Neural Networks for this classification task. Here is the definition from Wikipedia[4]:

*In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.*

*CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.*

*Convolutional networks were inspired by biological processes in which the connectivity pattern between neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.*

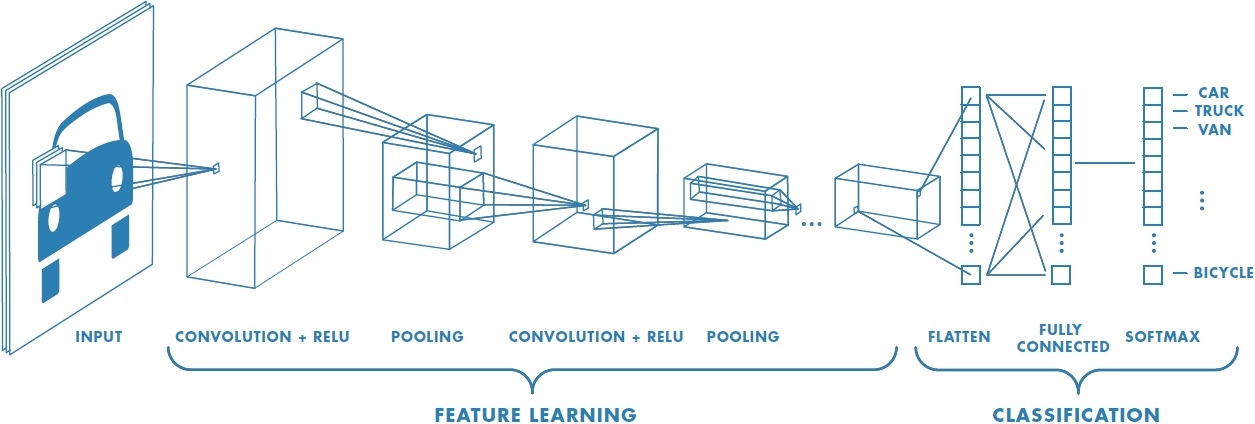


Figure 5 Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. [Source MathWorks](https://www.mathworks.com/discovery/convolutional-neural-network.html)

But to train a model from scratch we needed a lot of computational power and time. In order to reduce time we have used a model that was pretrained on the ImageNet Large Scale Visual Recognition Challenge [5].

## Xception Module for training

There is a lot of Neural Network Architectures [6] but for our pretrained model we will use Xception model. It’s a novel deep convolutional neural network architecture inspired by Inception, where Inception modules have been replaced with depthwise separable convolutions [7]. Apparently this architecture, slightly outperforms Inception V3 on the ImageNet dataset (which Inception V3 was designed for), and significantly outperforms Inception V3 on a larger image classification dataset comprising 350 million images and 17,000 classes.

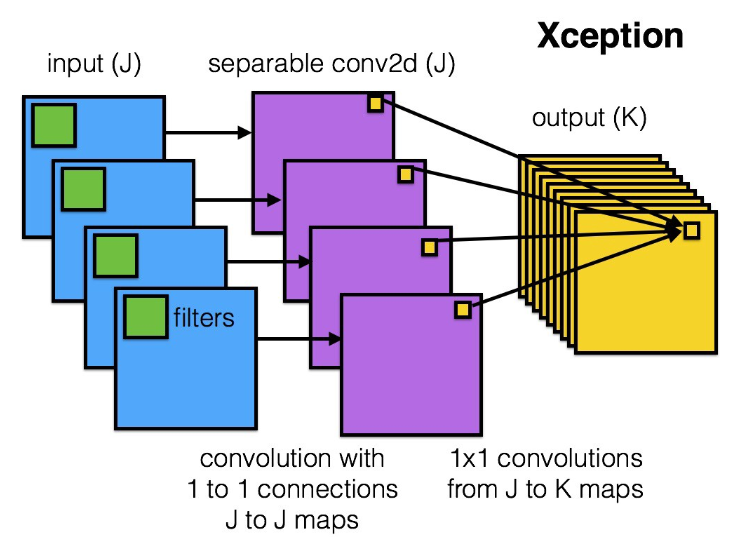


Figure 5 The Xception module

The architecture has 36 convolutional stages, making it close in similarity to a ResNet-34. But the model and code is as simple as ResNet and much more comprehensible than Inception V4 [8].

## Benchmark Model

Model now used by Cdiscount is KNN over TF-IDF [9] on the title and description of the items (with a lemmatisation to prepare the data). In production, they set up a threshold value for the level of confidence (i.e. the probability) of the predictions, in order to guarantee an error rate lower that some imposed value. They are able to classify two-thirds of the products with the imposed level of accuracy (classify products only on categories where they can reach more than 90% of accuracy). Cdiscount have been working on classifying products based on their images - best results so far obtained using Inception v3 and the accuracy around 70%.

In the competition the participants already tired different (single) models with the different resulting accuracy [10]. We will try to achive the same or better accuracy by using the following SE-ResNet-50 model as benchmark 70.91 accuracy on test set:

SE-ResNet-50

*SGD with Nesterov momentum (0.9) and batch\_size=256*

*finetuning about 7 epochs (about 16 hours per epoch on 2x1080)*

*train augmentation: 161x161 random crops + horizontal flips*

## Preprocessing data for pretrained Keras model

The size of data is big 60G - it's impossible to load all data in memory. It can cause a lot of trouble - the training time is already huge and choosing wrong loading model can add too many hours to the training process and iterations. Using SSD helped a lot at speeding the training process. We first decoded BSON file and write all images to the disc putting each image in respective category, 5270 in total. It takes more space but reduces the number of operations need to process images in our model, thus speeding up training time.

Here are some preprocessing steps used in this project:

* For this project only 10% of all images are used for convenience and possibility of replication.
* The use of 80/20 Training/Validation split. We can use a random split or a stratified split. The problem with the last is the data distribution - some classes have a lot of examples and other just a few. Sklearn can complain about the stratified split with 10000 values because some classes end up with just 1 product. So we use a simple split instead of stratified by removing the 'stratify=category\_ids' parameter.
* When using TensorFlow as backend, Keras CNNs require a 4D array as input, with shape (nb\_samples,rows,columns,channels). Tensorflow scales pixels between -1 and 1. Xception module has a build in normalization function – we need just to rescale the images.

Data augmentation and transformation:  
There is two ways in how to use data augmentation upsampling and downsampling. I'll try to do an upsampling from 180x180 to 224x224 and adding some data augmentation. But apparently it makes more sense to use dilated convolutions instead, since then the input is not interpolated, but it still maintains high resolution feature maps. The low-level layers can still stay the same and use the Imagenet weights as initialization since these are pretty much only simple filters (which for example react to edges, like Sobel). The mid- to high-level filters will probably have to change a lot, it may make sense to use dilated convolutions there. I'll also try downsampling by cropping from 180x180 to 160x160 as suggested by many posts in the forum.

Sources:  
<https://github.com/Cadene/pretrained-models.pytorch/issues/8>  
<https://www.reddit.com/r/MachineLearning/comments/52drsq/what_is_dilated_convolution/>  
<http://colah.github.io/posts/2014-12-Groups-Convolution/>

By looking at the images it can be tempting to desaturate photos (RGB -> NB). But apparently the models got stuck with only color/saturation augmentation enabled. The ones are doing much better only had random crop/scale and rotation. The prevalence of almost constant black or white backgrounds in most images means that adding variability to that requires much more learning than leaving it as it is. Another angle to explore is to see if disabling the random crop and just leaving horizontal flip and a bit of rotation does better, basically assuming that the preprocessing of the competition dataset leaves things relatively well centered and similarly scaled across train and test datasets. There are many classes with only a handful of images. For these classes data augmentation may be more important than for the classes with 80,000 images.

Keras Importance Sampling:   
Imbalance problem is present in this challenge - the distributions of the samples vary a lot. The use of importance sampling can help. Importance sampling focuses the computation to informative/important samples (by sampling mini-batches from a distribution other than uniform), thus accelerating the convergence. Also importance sampling has been successfully used to accelerate stochastic optimization in many convex problems (this method results in 30% faster training of a CNN for CIFAR10 than when using uniform sampling).

*Sources*:   
<http://idiap.ch/~katharas/importance-sampling/>  
<https://github.com/idiap/importance-sampling>  
["Sample Importance in Training Deep Neural Networks".](https://openreview.net/forum?id=r1IRctqxg)  
["A systematic study of the class imbalance problem in convolutional neural networks".](https://arxiv.org/pdf/1710.05381.pdf)  
["Biased Importance Sampling for Deep Neural Network Training".](https://arxiv.org/abs/1706.00043)

## MODEL TRAINING

In the training phase of the CNN where we are using Bottleneck Features [11] of a pre-trained network (Xception in our case). ) We only adding a 5270-element dense layer with Softmax activation function at the end (as we have 5270 categories) and also set only the last "block" of the network to be trainable and “freeze” the others (so it trains the classification layer and fine-tunes the last layer of the network only).

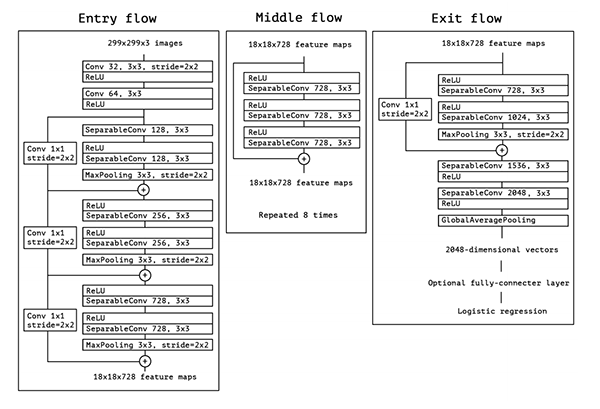


Figure : The Xception layer’s architecture.

We use SGD (stochastic gradient descent) [12] with momentum as our optimizer. Momentum is method which helps accelerate gradients vectors in the right directions, thus leading to faster converging [13]. It is one of the most popular optimization algorithms and many state-of-the-art models are trained using it. One of the important parameters of gradient descent is the learning rate. With a large learning rate gradient descent to converge faster but with lesser quality of weights, while small learning rate has lower converge speed but achieves better weight. Thus, we have changed the parameter along the training with certain decay factor. We start with large learning rate value and gradually decrease the learning rate as we iterate over our images by factor of 0.95.

We have built the CNV using Keras with Tensor Flow [14] backend as provided part of this submission. Keras already has a build in normalization function build on top of Tensorflow.

We also used drop-out of randomly selected units to increase the randomization of each stream from one layer to another, so each data going from one layer to another is randomly dropped with certain probability [15]. This will help prevent the over-fitting.

There is many methods for online batch selection for faster training of neural networks [16] but after trying different batch sizes, from 64 to 1024, the best performance was given by the batch size of 258 (86 x 3 GPUs). Training this kind of networks is complicated due to the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. To accelerate training time we used Batch Normalization already provided in Keras, it allows us to use much higher learning rates and be less careful about initialization [17].

# Conclusion

Training neural networks needs a large number of training examples, and our dataset has a lot of training examples, **but for some products there is only few examples so we will use Data Augmentation to increase the size of our training/validation sets by taking each training image and creating multiple random transformations around the bounding box of the object using ImageDataGenerator from Keras**. I'll use this technic only on the categories with a few images.

The original images are 180x180 so training on random crops of 160x160 is a quick way to do data augmentation (ImageNet models are often trained on crops of 224x224). I will start with 1-2 epochs.

I'll use Xception with a 5270-neuron Softamx classification layer on Keras (TensorFlow), with the batch size 256 at first, training/validation split is 80%-20%, input image size 160x160 (downsampling). It will takes around 4-6 hours to do a single epoch with 8 workers in model.fit\_generator() on EC2 p2.xlarge. This will use the BSON generator posted in competition kernels.

We have successfully used transfer learning to classify such large and imbalanced data set. It’s really impressive to achive this results with the limited ressources and time. By using transfer learning we cut our training time by order of 5-10 times!

Our localization algorithm shows 70% true positive and approximately 20%

false positive results.

We may need to improve the false positive by changing the threshold or clustering

the decision points.

Even though the results are promising. One of the next step of this project is to improve the false positive rate for the objects that actually does not exist.

This can be improved by using longer training sequence and more randomization.

**But still there is a lot of room for improvement, here is some points to investigate:**

Ensemble of Deep Convolutional Neural Networks

////

<https://pure.tugraz.at/portal/files/7666387/opitz_accv_16.pdf>

The use of hard samples.

Based on the prediction scores, it can identify which training (and testing) samples are difficult. Training and testing datasets contain an overwhelming number of easy examples and a small number of hard examples (just a few examples per class). Automatic selection of these hard examples can make training more effective and efficient. We can speedup training with fewer samples. And use the data augmentation only on these examples.

Learning Rate Control.

It is observed that the models trained by SGD are sensitive to learning rates and good learning rates are problem specific. There is an algorithm to automatically learn learning rates using neural network based actor-critic methods from deep reinforcement learning (RL). In particular, we train a policy network called actor to decide the learning rate at each step during training, and a value network called critic to give feedback about quality of the decision (e.g., the goodness of the learning rate outputted by the actor) that the actor made. [https://arxiv.org/abs/1705.11159]

Accelerating Very Deep Convolutional Networks by Using Nonlinearity.

While almost all methods mainly focus on optimizing one or two layers, this nonlinear method enables an asymmetric reconstruction that reduces the rapidly accumulated error when multiple (e.g., >=10) layers are approximated. For the widely used very deep VGG-16 model, this method achieves a whole-model speedup of 4x with merely a 0.3% increase of top-5 error in ImageNet classification. The 4x accelerated VGG-16 model also shows a graceful accuracy degradation for object detection when plugged into the Fast R-CNN detector.

[https://arxiv.org/abs/1505.06798]

Use Enseble of CNNs

Pomp from here:  
[VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION.](https://arxiv.org/pdf/1409.1556.pdf)  
[AN ANALYSIS OF DEEP NEURAL NETWORK MODELS FOR PRACTICAL APPLICATIONS.](https://arxiv.org/pdf/1605.07678.pdf)

[Distilling the Knowledge in a Neural Network.](https://www.cs.toronto.edu/~hinton/absps/distillation.pdf)

POSSIBLE Exploration

For a moment I'll stick to the single model but it's worth investigate the use of an ensemble of models.

[Distilling the Knowledge in a Neural Network.](https://www.cs.toronto.edu/~hinton/absps/distillation.pdf)   
"... We introduce a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel."

REFERENCES

[1] [Cdiscount’s Image Classification Challenge on Kaggle](https://www.kaggle.com/c/cdiscount-image-classification-challenge)

[2] [An Analysis of Deep Neural Network Models for Practical Applications](https://arxiv.org/abs/1605.07678)

[3] [A Gentle Introduction to Transfer Learning for Deep Learning](https://machinelearningmastery.com/transfer-learning-for-deep-learning/)

[4] https://en.wikipedia.org/wiki/Convolutional\_neural\_network

[5] http://www.image-net.org/challenges/LSVRC

[6] https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba

[7] <https://arxiv.org/abs/1610.02357>  
[8] <https://towardsdatascience.com/neural-network-architectures-156e5bad51ba>

[9] <https://www.sciencedirect.com/science/article/pii/S1877705814003750>

[10] <https://www.kaggle.com/c/cdiscount-image-classification-challenge/discussion/41652>

[11] <https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html>

[12] https://en.wikipedia.org/wiki/Stochastic\_gradient\_descent

[13] <https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d>

[14] https://www.tensorflow.org

[15] <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>

[16] <https://arxiv.org/abs/1511.06343>

[17] https://arxiv.org/abs/1502.03167